(19) JAPANESE PATENT OFFICE

PATENT ABSTRACTS OF JAPAN

(11) Publication number: **09128246 A**

(43) Date of publication of application: **16.05.97**

(51) Int. Cl **G06F 9/45**

(21) Application number: **07281778**

(22) Date of filing: **30.10.95**

(71) Applicant: **FUJITSU LTD**

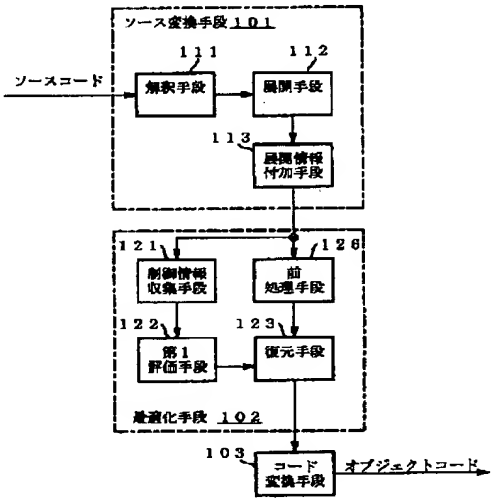
(72) Inventor: **HAYASHI MASAKAZU**
BABA SADAICHI

(54) COMPILER DEVICE

(57) Abstract:

PROBLEM TO BE SOLVED: To obtain an object code whose processing efficiency is high by evaluating the propriety of the in-line development of a pertinent function based on static control information, outputting evaluated results for respective in-line developing parts and restoring the pertinent in-line developing part into an original function call word in accordance with the evaluated result.

SOLUTION: The control information collection means 121 of an optimizing means 102 analyzes an intermediate text string obtained by a source conversion means 101 and collects static control information on the flow of control at the time of execution and on the flow of data. A first evaluation means 122 evaluates the propriety of the in-line development of the pertinent function in a developing means 112, and outputs the evaluated results for the respective in-line developing parts. A restoration means 123 restores the pertinent in-line developing part into the original function call word. Thus, the object code whose processing efficiency is high can be obtained.



(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平9－128246

(43)公開日　平成９年(1997)５月16日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/45			Ｇ０６Ｆ　9/44	３２２Ｆ	
				３２２Ｍ	

審査請求　未請求　請求項の数５　ＯＬ　（全 19 頁）

(21)出願番号　特願平7－281778

(22)出願日　平成７年(1995)10月30日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　林　正和
神奈川県川崎市中原区上小田中1015番地
富士通株式会社内

(72)発明者　馬場　貞一
神奈川県川崎市中原区小杉町１丁目403番地　株式会社富士通ソーシアルサイエンスラボラトリ内

(74)代理人　弁理士　古谷　史旺　（外１名）

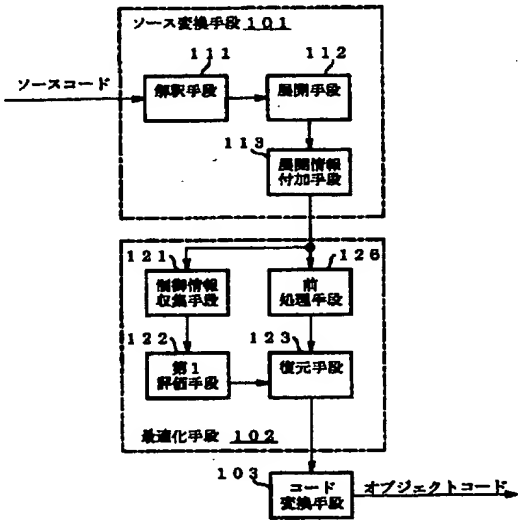
(54)【発明の名称】　コンパイラ装置

(57)【要約】

【課題】　インライン展開後に得られる情報に応じて、各関数のインライン展開を行うコンパイラ装置を提供する。

【解決手段】　ソース変換手段１０１で得られた中間テキストを最適化手段１０２が最適化し、コード変換手段１０３がオブジェクトコードに変換するコンパイラ装置において、ソース変換手段１０１は、ソースコードを中間テキスト列に変換する解釈手段１１１と、関数呼び出し言をインライン展開する展開手段１１２と、インライン展開を示す展開情報を付加する展開情報付加手段１１３とを備え、最適化手段１０２は、中間テキスト列を解析して静的制御情報を収集する制御情報収集手段１２１と、静的制御情報に基づいて、各関数をインライン展開したことの当否を評価する第１評価手段１２２と、評価結果に応じて、該当するインライン展開部分を元の関数呼び出し言に復元する復元手段１２３とを備える。

請求項１および請求項３のコンパイラ装置の原理ブロック図



1

【特許請求の範囲】

【請求項1】　入力されたソースコードをソース変換手段によって中間テキストに変換して最適化手段の処理に供し、この最適化手段による処理結果をコード変換手段によってオブジェクトコードに変換するコンパイラ装置において、

前記ソース変換手段は、

ソースコードのそれぞれを解釈して対応する中間テキスト言に変換する解釈手段と、

前記中間テキストに含まれる関数呼び出し言を該当する関数本体で置換することで関数のインライン展開を行う展開手段と、

前記展開手段でインライン展開された部分の範囲を示す展開情報を中間テキストに付加して出力する展開情報付加手段とを備えた構成であり、

前記最適化手段は、

前記ソース変換手段で得られた中間テキスト列を解析し、実行時の制御の流れおよびデータの流れに関する静的制御情報を収集する制御情報収集手段と、

前記静的制御情報に基づいて、前記展開情報で示された関数本体部分のそれぞれについて、該当する関数をインライン展開したことの当否を評価して、インライン展開部分ごとに評価結果を出力する第1評価手段と、

前記評価結果に応じて、該当するインライン展開部分を元の関数呼び出し言に復元する復元手段とを備えた構成であることを特徴とするコンパイラ装置。

【請求項2】　入力されたソースコードをソース変換手段によって中間テキストに変換して最適化手段の処理に供し、この最適化手段による処理結果をコード変換手段によってオブジェクトコードに変換するコンパイラ装置において、

前記ソース変換手段は、

ソースコードのそれぞれを解釈して対応する中間テキスト言に変換する解釈手段と、

前記中間テキストに含まれる関数呼び出し言を該当する関数本体で置換することで関数のインライン展開を行う展開手段と、

前記展開手段でインライン展開された部分の範囲を示す展開情報を中間テキストに付加して出力する展開情報付加手段とを備えた構成であり、

前記最適化手段は、

前記変換手段で得られた中間テキスト列を擬似的に実行して得られる各中間テキストの実行回数を含む動的制御情報の入力を受け付ける入力受付手段と、

前記動的制御情報に基づいて、前記展開情報で示された関数本体部分のそれぞれについて、該当する関数をインライン展開したことの当否を評価して、インライン展開部分ごとに評価結果を出力する第2評価手段と、

前記評価結果に応じて、該当するインライン展開部分を元の関数呼び出し言に復元する復元手段とを備えた構成

2

であることを特徴とするコンパイラ装置。

【請求項3】　請求項1または請求項2に記載のコンパイラ装置において、

最適化手段は、

ソース変換手段で得られた中間テキスト列に対して、インライン展開部分の範囲の境界を保存しながら最適化処理を行い、処理結果をインライン展開の当否を評価する処理に供する前処理手段を備えた構成であることを特徴とするコンパイラ装置。

【請求項4】　請求項3に記載のコンパイラ装置において、

最適化手段は、

復元手段による処理結果に対して、インライン展開部分と他の部分とを同等のものとして最適化処理を行う後処理手段を備えた構成であることを特徴とするコンパイラ装置。

【請求項5】　請求項1または請求項2に記載のコンパイラ装置において、

コード変換手段は、入力された中間テキスト列に含まれた展開情報に基づいて、インライン展開された部分に対応するオブジェクトコード列それぞれに、インライン展開された部分である旨を示すコード情報を付加するコード付加手段を備えた構成であることを特徴とするコンパイラ装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、関数をインライン処理して最適化を図るコンパイラ装置に関するものである。近年、コンピュータシステムに対する高速化の要求がより高まっており、これに応じて様々の種類のプロセッサが開発されている。これに伴って、それぞれのプロセッサの特徴を活かして、更なる高速処理を実現するために、コンパイラ装置の最適化機能に対する要求も高度化している。コンパイラ装置における最適化処理においては、様々な機能が利用されており、関数呼び出しの検出に応じて、該当する関数の本体を呼び出し元に展開するインライン展開機能もその一つである。このインライン展開機能により、関数呼び出しのためのオーバーヘッドの解消や分岐命令の実行回数の削減による直接的な実行速度の向上効果が得られる。また、インライン展開によって明らかにされた引数などの情報に基づいて、更に最適化が可能となることによる間接的な実行速度の向上効果が得られる。このため、インライン展開機能は、コンパイラ装置に必須の機能となっている。

【0002】

【従来の技術】上述したように、関数のインライン展開により、大きな最適化効果が得られる反面、インライン展開によって、オブジェクトコードそのものが大幅に増大し、プログラム格納のためのディスクスペースを圧迫したり、命令キャッシュミスが増大するなどの不利益が

ある。また、必要なレジスタ数の増大を招いたり、翻訳時間が増大したりといったデメリットが生じることも知られている。

【０００３】このため、従来のコンパイラ装置においては、プログラム全体について、コンパイルオプションにより、コンパイラ装置に対する様々な指示とともにインライン展開の要否を指定する場合がある。図１１に、従来のコンパイラ装置の構成例を示す。図１１において、フロントエンド部４０１は、入力されたソースプログラムを解釈して中間テキストに変換し、インライン展開部４０２は、コンパイル制御部４０３からの指示に応じて、関数呼び出しに対応する関数を表す中間テキストをインライン展開し、最適化処理部４０４による処理に供する構成となっている。

【０００４】また、最適化処理部４０４による処理結果は、バックエンド部４０５およびコード出力部４０６による処理に供され、最終的なオブジェクトプログラムが作成される。このとき、コンパイル制御部４０３は、プログラム開発者の判断で指定されたコンパイルオプションに応じて、インライン展開部４０２の動作を制御しており、オプションなどで指定されたインライン可能なオブジェクトサイズ以下の全ての関数がインライン展開対象として指示される。

【０００５】このため、オブジェクトサイズが小さい関数は、呼び出し回数にかかわらずにインライン展開されるのに対して、オブジェクトサイズの大きい関数は、呼び出し回数が多くてもインライン展開されないため、インライン展開による効果が充分に得られない場合があった。このような問題点を解決するために、様々な技法が提案されている。

【０００６】例えば、特開平３－９９３３０「手続きインライン展開方式」で開示された技法は、コンパイル処理の中間段階で得られる中間テキストに基づいて、各関数の静的な参照回数を計数し、この参照回数に応じて、それぞれの関数をインライン展開するか否かを判定するものである。また、特開平６－２０２８７５「インライン展開による最適化を行うコンパイラ」で開示された技法は、中間テキストに基づいて、各関数の実行回数を予測して重み付けを行い、この重みに応じて、インライン展開を行うか否かを判定するものである。

【０００７】更に、特開平１－１１８９３１「プログラム変換方式」で開示された技法は、全ての関数呼び出しを保存した仮の実行プログラムを作成し、この仮の実行プログラムを実行した際に得られる動的な関数の呼び出し回数やループ回数に基づいて、各関数をインライン展開するか否かを判定するものである。

【０００８】これらの技法を適用することにより、コンパイル制御部４０３において、個々の関数の呼び出し回数を考慮してインライン展開部４０２を制御することが可能となり、インライン展開による最適化効果を高めることができる。一方、特開平５－７３３３５「プログラム自動インライン展開方式」で開示された技法は、ソースプログラムを解析することにより、他の関数を呼び出していない関数を判別し、該当する関数をインライン展開するものであり、上述したような静的な参照回数や動的な参照回数を計数する手間を省いて、翻訳時間の短縮を図っている。

【０００９】

【発明が解決しようとする課題】ところで、上述した従来の技法は、全て、インライン展開に先だって、その要否を判定するものであり、インライン展開後に明らかとなる情報は一切考慮されていない。例えば、インライン展開によって拡大した各変数の生存期間は、インライン展開以前に予測することは困難である。一方、実行可能なオブジェクトプログラムを効率良く作成するためには、各時点において必要なレジスタの数を一定数以下に抑えておく必要があるから、この拡大された生存期間に基づいて各変数にレジスタを割付けなければならない。さもないと、不要なレジスタの内容をメモリに退避するためのコード（スピルコード）が出力され、性能低下を招くからである。

【００１０】しかしながら、従来のコンパイラ装置においては、インライン展開後の変数の生存期間の拡大は考慮されていないから、ある関数をインライン展開したことによって、バックエンド部４０５における割付処理などが厳しく制約されてしまい、最適化効率が低下する場合がある。

【００１１】このように、インライン展開処理後には、それまで関数名の陰に隠されていた様々な情報が明らかにされ、これらの情報が、後の最適化処理による効果を大きく左右するのである。本発明は、インライン展開後に得られる情報に応じて、各関数のインライン展開を行うコンパイラ装置を提供することを目的とする。

【００１２】

【課題を解決するための手段】図１は、請求項１および請求項３のコンパイラ装置の原理ブロック図である。請求項１の発明は、入力されたソースコードをソース変換手段１０１によって中間テキストに変換して最適化手段１０２の処理に供し、この最適化手段１０２による処理結果をコード変換手段１０３によってオブジェクトコードに変換するコンパイラ装置において、ソース変換手段１０１は、ソースコードのそれぞれを解釈して対応する中間テキスト言に変換する解釈手段１１１と、中間テキストに含まれる関数呼び出し言を該当する関数本体で置換することで関数のインライン展開を行う展開手段１１２と、展開手段１１２でインライン展開された部分の範囲を示す展開情報を中間テキストに付加して出力する展開情報付加手段１１３とを備えた構成であり、最適化手段１０２は、ソース変換手段１０１で得られた中間テキスト列を解析し、実行時の制御の流れおよびデータの流

れに関する静的制御情報を収集する制御情報収集手段１２１と、静的制御情報に基づいて、展開情報で示された関数本体部分のそれぞれについて、該当する関数をインライン展開したことの当否を評価して、インライン展開部分ごとに評価結果を出力する第１評価手段１２２と、評価結果に応じて、該当するインライン展開部分を元の関数呼び出し言に復元する復元手段１２３とを備えた構成であることを特徴とする。

【００１３】請求項１の発明は、ソース変換手段１０１において、展開手段１１２が、解釈手段１１１によって得られた関数呼び出し言を該当する関数本体に置換したときに、展開情報付加手段１１３により、インライン展開された中間テキストの範囲を示す展開情報を付加し、最適化手段１０２の処理に供している。したがって、最適化手段１０２において、第１評価手段１２２および復元手段１２３は、上述した展開情報によってインライン展開された部分を知ることができる。

【００１４】これにより、評価手段１２２が、制御情報収集手段１２１で得られた静的制御情報に基づいて、各インライン展開部分についての評価を行うことができ、また、この評価結果に応じて、復元手段１２３が動作することにより、インライン展開したことが妥当でないとされたインライン展開部分を元の関数呼び出し言に戻すことができる。以下、インライン展開された関数本体を元の関数呼び出し言に戻す操作を、「インライン」に対して「アウトライン」と称する。

【００１５】上述したようにして、インライン展開後に明らかとなる情報を考慮して、インライン展開による最適化の効果を判断し、関数呼び出しの方が適切であるものについてはアウトラインすることにより、有効なインライン展開部分のみを含んだ中間テキスト列をコード変換手段１０３に送出することが可能であるから、インライン展開による最適化効果を十分に引き出すことができる。

【００１６】図２は、請求項２および請求項４、５のコンパイラ装置の原理ブロック図である。請求項２の発明は、入力されたソースコードをソース変換手段１０１によって中間テキストに変換して最適化手段１０２の処理に供し、この最適化手段１０２による処理結果をコード変換手段１０３によってオブジェクトコードに変換するコンパイラ装置において、ソース変換手段１０１は、ソースコードのそれぞれを解釈して対応する中間テキスト言に変換する解釈手段１１１と、中間テキストに含まれる関数呼び出し言を該当する関数本体で置換することで関数のインライン展開を行う展開手段１１２と、展開手段１１２でインライン展開された部分の範囲を示す展開情報を中間テキストに付加して出力する展開情報付加手段１１３とを備えた構成であり、最適化手段１０２は、変換手段１０１で得られた中間テキスト列を擬似的に実行して得られる各中間テキストの実行回数を含む動的制御情報の入力を受け付ける入力受付手段１２４と、動的制御情報に基づいて、展開情報で示された関数本体部分のそれぞれについて、該当する関数をインライン展開したことの当否を評価して、インライン展開部分ごとに評価結果を出力する第２評価手段１２５と、評価結果に応じて、該当するインライン展開部分を元の関数呼び出し言に復元する復元手段１２３とを備えた構成であることを特徴とする。

【００１７】請求項２の発明は、請求項１と同様に、ソース変換手段１０１の解釈手段１１１、展開手段１１２および展開情報付加手段１１３の動作により、インライン展開された部分の範囲を示す展開情報を含んだ中間テキスト列が得られ、最適化手段１０２の処理に供される。この場合は、入力受付手段１２４を介して、各中間テキストの実行回数を正当に評価した動的制御情報が、第２評価手段１２５の処理に供され、これにに基づいて、各インライン展開部分についての評価が行われるから、インライン展開したことの当否をより正当に評価することができる。

【００１８】したがって、この評価結果に応じて、復元手段１２３が動作することにより、インライン展開が真に有効なもののみをインライン展開した状態で保存してコード変換手段１０３の処理に供することができる。請求項３の発明は、請求項１または請求項２に記載のコンパイラ装置において、最適化手段１０２は、ソース変換手段１０１で得られた中間テキスト列に対して、インライン展開部分の範囲の境界を保存しながら最適化処理を行い、処理結果をインライン展開の当否を評価する処理に供する前処理手段１２６を備えた構成であることを特徴とする。

【００１９】請求項３の発明は、前処理手段１２６による処理結果を第１評価手段１２２または第２評価手段１２５の処理に供することにより、限定された最適化処理による効果を考慮して、インライン展開の当否を評価することができる。なお、この前処理手段１２６において、インライン展開部分の境界を跨いだ中間テキストの移動を禁止すれば、インライン展開部分の境界を保存した最適化結果を得ることができる。

【００２０】請求項４の発明は、請求項３に記載のコンパイラ装置において、最適化手段１０２は、復元手段１２３による処理結果に対して、インライン展開部分と他の部分とを同等のものとして最適化処理を行う後処理手段１２７を備えた構成であることを特徴とする。請求項４の発明は、復元手段１２３による処理結果の入力に応じて、後処理手段１２７が動作することにより、インライン展開部分の境界を跨いだ中間テキストの移動を含んだ最大限の最適化を中間テキストの施すことができ、実行性能の向上を図ることができる。

【００２１】請求項５の発明は、請求項１または請求項２に記載のコンパイラ装置において、コード変換手段１

０３は、入力された中間テキスト列に含まれた展開情報に基づいて、インライン展開された部分に対応するオブジェクトコード列それぞれに、インライン展開された部分である旨を示すコード情報を付加するコード付加手段１３１を備えた構成であることを特徴とする。

【００２２】請求項５の発明は、コード変換手段１０３のコード付加手段１３１が、中間テキストに残された展開情報に基づいて動作することにより、インライン展開された部分に対応するオブジェクトコードの範囲を示す情報として、例えばコメント文などをオブジェクトコードの中に挿入し、デバッグ作業に供することができる。

【００２３】

【発明の実施の形態】以下、図面に基づいて、本発明の実施例について詳細に説明する。

【００２４】図３は、本発明のコンパイラ装置の実施例構成図である。図３に示したコンパイラ装置は、図１１に示したコンパイラ装置のインライン展開部４０２と最適化処理部４０４とに代えてインライン展開部２１０と最適化前処理部２２０および最適化後処理部２３０とを備えた構成となっている。このコンパイラ装置において、インライン展開部２１０は、コンパイル制御部４０３からの指示に応じて、解釈手段１１１に相当するフロントエンド部４０１で得られた中間テキスト列についてインライン展開処理を行い、この処理結果を最適化前処理部２２０の処理に供する構成となっている。

【００２５】また、最適化前処理部２２０による処理結果に基づいて、アウトライン判定部２４０およびアウトライン処理部２５０が動作し、このアウトライン処理部２５０による処理結果が、最適化後処理部２３０、バックエンド部４０５およびコード出力部４０６の処理に供され、オブジェクトプログラムが作成される構成となっている。

【００２６】つまり、このコンパイラ装置においては、図４に示すように、フロントエンド部４０１により中間テキストが作成され（ステップ３０１）、コンパイラオプションに応じて、コンパイル制御部４０３がインライン展開を指示すると（ステップ３０２の肯定判定）、インライン展開部２１０により、全ての関数呼び出しを対象として一律にインライン展開処理が行われる（ステップ３０３）。

【００２７】その後、最適化前処理部２２０による最適化処理が行われ（ステップ３０４）、この処理結果に応じて、アウトライン判定部２４０とアウトライン処理部２５０とにより、上述したインライン展開処理部２１０によって展開された関数のうち、不適切なものが元の関数呼び出しに戻される（ステップ３０５、３０６）。このようにして得られた中間テキスト列に対して、最適化後処理部２３０により再度の最適化の試みが行われ（ステップ３０７）、次いで、ステップ３０８において、オブジェクトプログラムの作成処理が行われる。

【００２８】一方、上述したステップ３０２の否定判定の場合は、最適化前処理部２２０および最適化後処理部２３０により、従来と同様の最適化処理を行い（ステップ３０９）、その後、処理を終了すればよい。図３に示したインライン展開部２１０において、受付処理部２１１は、未処理の中間テキストから関数呼び出しを示すテキスト（以下、関数呼び出し言と称する）を抽出し、サイズ判定部２１２に送出する構成となっている。

【００２９】また、このインライン展開部２１０において、疑似言挿入部２１３および展開処理部２１４は、上述したサイズ判定部２１２による判定結果に応じて動作し、受付処理部２１１で抽出された関数呼び出し言を受け取って、後述する所定の疑似言と関数本体の中間テキスト列とに変換する構成となっている。ここで、疑似言挿入部２１３は、図５に示すように、インライン展開された関数の先頭および末尾に、それぞれインライン開始言（*BeginInline）とインライン終了言（*EndInline）とを挿入すればよい。

【００３０】また、図３に示した疑似言修飾部２１５は、疑似言挿入部２１３が挿入した疑似言と元の関数呼び出し言との対応関係を示す関数情報および２つの疑似言の対応関係を示す対応情報を作成し、上述したインライン開始言およびインライン終了言のオペランドとして付加する構成となっている。この疑似言修飾部２１５は、例えば、展開処理部２１４から元の関数呼び出し言や展開された中間テキストの数を受け取って、これらを示す関数情報を作成し、また、疑似言挿入部２１３から上述した２つの疑似言をアドレスを受け取って、これらをそのまま対応情報として、それぞれ対となる疑似言のオペランドに付加すればよい。

【００３１】図６に、インライン展開動作を表す流れ図を示す。受付処理部２１１は、未処理の中間テキストを１言ずつ順次に読み込んでいき（ステップ３１１）、関数呼び出し言を読み込んだときに、ステップ３１２の肯定判定として、サイズ判定部２１２に該当する関数呼び出し言を通知する。この通知に応じて、サイズ判定部２１２は、該当する関数本体を見つけ出し、存在すればその関数本体の中間テキスト列を計数し（ステップ３１３）、この計数値が所定の閾値以下であれば、ステップ３１４の肯定判定として、疑似言挿入部２１３を起動するとともに、該当する関数呼び出し言を展開処理部２１４に通知する。

【００３２】これに応じて、まず、疑似言挿入部２１３は、関数呼び出し言に代えて、上述したインライン開始言を挿入し（ステップ３１４）、展開処理部２１４は、このインライン開始言に続いて、該当する関数本体の中間テキスト列を未処理の中間テキスト列に挿入する（ステップ３１５）。このときに、必要であれば、引数の受け渡しのための中間テキストも挿入する。次いで、疑似言挿入部２１３は、ステップ３１６において、上述した

インライン終了言を未処理の中間テキストとして挿入すればよい。

【００３３】また、疑似言修飾部２１５は、ステップ３１７において、上述した関数情報および対応情報を作成し、ステップ３１４およびステップ３１６で挿入した疑似言のオペランドとして付加すればよい。一方、ステップ３１２の否定判定の場合は、該当する中間テキストをそのまま処理済みの中間テキストとして、後述するステップ３１８に進めばよい。

【００３４】その後、受付処理部２１１は、未処理の中間テキストが残っているか否かを判定し（ステップ３１８）、肯定判定の場合は、ステップ３１１に戻って新しい中間テキストについての処理を行い、否定判定の場合は、インライン展開処理を終了すればよい。このように、受付処理部２１１およびサイズ判定部２１２による判定結果に応じて、疑似言挿入部２１３と展開処理部２１４とが動作することにより、展開手段１１２と展開情報付加手段１１３との機能を実現し、展開可能な全ての関数呼び出しをインライン展開するとともに、インライン展開された部分を示す展開情報として上述した２つの疑似言を付加することができる。

【００３５】上述したようにして得られた中間テキストは、図３に示すように、最適化前処理部２２０に送出され、後述する基本的な最適化処理が施される。図３に示した最適化前処理部２２０において、基本処理部２２１は、請求項３で述べた前処理手段１２６に相当するものであり、上述した疑似言で挟まれた範囲内での中間テキストの移動や共通式の除去および演算の強さの縮小などの基本的な最適化を行う構成となっている。

【００３６】また、データフロー収集部２２２および制御フロー収集部２２３は、請求項１で述べた制御情報収集手段１２１に相当するものであり、インライン展開部２１０を介して受け取った中間テキスト列を解析し、各変数の生存期間やループ範囲などを示す制御情報を収集し、アウトライン判定部２４０の処理に供する構成となっている。

【００３７】このアウトライン判定部２４０において、ループ判定部２４１は、インライン開始言を検出することに、そのアドレスと制御情報として受け取ったループ範囲とを照合し、該当するインライン開始言がループ内に含まれているか否かを判定する構成となっている。この判定結果に応じて、サイズ判定部２４２とレジスタ数判定部２４３とが動作し、該当するインライン開始言と対応するインライン終了言との間に挟まれた部分について、そのサイズが所定の閾値以下であるか否か、および生存中の変数の数が所定の閾値以下であるか否かをそれぞれ判定する構成となっている。

【００３８】また、候補決定部２４４は、これらの判定結果に応じて、アウトライン候補を決定し、該当するインライン開始言を候補リスト２４５に順次に格納し、アウトライン処理部２５０の処理に供する構成となっている

図７に、アウトライン判定動作を表す流れ図を示す。まず、ループ判定部２４１は、未処理の中間テキストを順次に読み出して（ステップ３２１）、インライン開始言であるか否かを判定し（ステップ３２２）、肯定判定の場合は、制御情報に基づいて、該当するインライン開始言がループ範囲に含まれているか否かを判定する（ステップ３２３）。

【００３９】ステップ３２３の否定判定の場合に、サイズ判定部２４２は、該当するインライン開始言と対応するインライン終了言との間の中間テキストの数を計数し（ステップ３２４）、この計数値Ntが所定の閾値Ths を超えているか否かを判定する（ステップ３２５）。

【００４０】このステップ３２５の否定判定の場合は、レジスタ数判定部２４３が、該当するインライン開始言と対応するインライン終了言との間の中間テキスト列について、上述した制御情報に基づいて同時に生存している変数の数の最大値Maxvを求め（ステップ３２６）、この最大値Maxvが、所定の閾値Thv を超えているか否かを判定する（ステップ３２７）。

【００４１】なお、上述した中間テキスト数の閾値Thsおよび変数の数の閾値Thv は、それぞれ情報処理装置の性能およびプロセッサに備えられているレジスタの数に応じて予め決定しておき、例えば、コンパイル制御部４０３により、それぞれサイズ判定部２４２およびレジスタ数判定部２４３の処理に供すればよい。上述したステップ３２７の肯定判定あるいは、上述したステップ３２５の肯定判定の場合に、候補決定部２４４は、該当するインライン開始言を候補リスト２４５に格納する（ステップ３２８）。

【００４２】その後、未処理の中間テキストが残っているか否かを判定し（ステップ３２９）、肯定判定の場合は、上述したステップ３２１に戻って新しい中間テキストについての処理を行えばよい。また、上述したステップ３２２の否定判定の場合およびステップ３２３の肯定判定の場合は、上述したアウトライン判定処理をスキップし、そのままステップ３２９に進めばよい。

【００４３】このように、ループ判定部２４１からの指示に応じて、サイズ判定部２４２およびレジスタ数判定部２４３が動作することにより、請求項１で述べた第１評価手段１２２の機能を実現し、中間テキストに含まれる全てのインライン関数について、インライン展開したことの当否を判定することができる。また、ステップ３２３の判定処理において、ループの次元を考慮に入れることも可能である。

【００４４】また、このようにして開始言を順次にチェックしていき、全ての中間テキストについての処理が終了したときに、ステップ３２９の否定判定としてアウトライン判定処理を終了し、その結果をアウトライン処理

部２５０の処理に供すればよい。図３において、アウトライン処理部２５０は、候補検出部２５１が、上述した候補リスト２４５に基づいて、入力される中間テキストから該当するインライン開始言を検出し、これに応じて、削除処理部２５２と置換処理部２５３とが動作する構成となっている。

【００４５】この削除処理部２５２は、上述した候補検出部２５１を介して中間テキスト列を受け取り、アウトライン候補として検出されたインライン開始言から対応するインライン終了言までの中間テキストを削除する構成となっている。また、置換処理部２５３は、該当するインライン開始言にオペランドとして付加された関数情報に基づいて、インライン開始言およびインライン終了言を元の関数呼び出し言で置換する構成となっている。

【００４６】図８に、アウトライン処理動作を表す流れ図をそれぞれ示す。まず、候補検出部２５１は、中間テキストを１言ずつ読み出して、インライン開始言であるか否かを判定し（ステップ３３１、３３２）、インライン開始言である場合（ステップ３３２の肯定判定）は、該当するインライン開始言が上述した候補リスト２４５に格納されているか否かを判定する（ステップ３３３）。

【００４７】このステップ３３３の肯定判定の場合は、削除処理部２５１が動作し、該当するインライン開始言と対応するインライン終了言とで挟まれた中間テキストを削除する（ステップ３３４）。次に、復元処理部２５３が動作し、該当するインライン開始言に付加された関数情報からもとの関数呼び出し言を得て、この関数呼び出し言を例えば上述したインライン開始言の直後に挿入し（ステップ３３５）、更に、該当するインライン開始言およびこれに対応するインライン終了言を削除する（ステップ３３６）。

【００４８】このステップ３３６の終了後および上述したステップ３３２およびステップ３３３の否定判定の場合は、未処理の中間テキストがあるか否かを判定し（ステップ３３７）、肯定判定の場合は、ステップ３３１に戻って新しい中間テキストについての処理を行えばよい。このように、候補検出部２５１からの指示に応じて、削除処理部２５２と復元処理部２５３とが動作することにより、請求項１で述べた復元手段１２３の機能を実現し、中間テキストの中からインライン展開の効果が否定された部分を抽出し、元の関数呼び出しに戻すことができる。

【００４９】また、このようにして順次に全てのアウトライン候補を元の関数呼び出し言に戻した後に、ステップ３３７の否定判定に応じて、アウトライン処理を終了し、得られた処理結果を最適化後処理部２３０の処理に供すればよい。この最適化後処理部２３０は、請求項４で述べた後処理手段１２７に相当するものであり、インライン開始言およびインライン終了言を跨いだ中間テキストの移動を含めて、アウトライン処理後の中間テキストに対して、再度の最適化処理を行い、バックエンド部４０５に中間テキスト列を送出すればよい。

【００５０】その後は、バックエンド部４０５とコード出力部４０６とが、従来と同様に動作することにより、最適化された中間テキストに対応するオブジェクトコードが得られる。この場合は、上述したように、全ての関数呼び出しをインライン展開した後に、展開によって明らかにされた情報を踏まえてインライン展開の当否を判定し、展開が展開が効果的であるとされた関数呼び出しのみを最適化後の中間テキストに保存することができる。

【００５１】したがって、インライン展開後に明らかにされる情報を考慮して、レジスタ割付処理などへの負担を軽減し、真に効果的なインライン展開結果のみを含んだ中間テキストに対応するオブジェクトコードを得ることができるから、コンパイラ装置のインライン展開機能により、充分な最適化効果を得ることが可能となる。

【００５２】例えば、利用頻度の高い作業を実行するためのアプリケーションプログラムのコンパイルに利用すれば、得られたオブジェクトコードによる処理効率を向上することが可能となる。特に、近年のプロセッサの開発競争においては、レジスタ本数や演算器の数などが多様化していることを考えれば、インライン展開後に詳細となる情報を利用して、インライン展開する関数を選択することが可能であることによる最適化効果は非常に大きく、それに必要とされるコンパイル時間の増大などのデメリットを十分に補うことができる。

【００５３】更に、アウトライン判定処理において、各インライン展開部の実際の実行回数に応じて、アウトラインするか否かを判定することもできる。図９に、請求項２のコンパイラ装置の実施例構成図を示す。図９において、請求項２のコンパイラ装置は、図３に示したコンパイラ装置に、プロファイル抽出部２６１と実行処理部２６２とを付加し、また、アウトライン判定部２５０に代えてアウトライン判定部２７０を備えた構成となっている。

【００５４】この場合は、コンパイル制御部４０３は、最適化前処理部２２０による処理結果を実行制御部２６２に送出してその実行を指示し、これに応じて、実行処理部２６２が中間テキスト列を擬似的に実行したときに、プロファイル抽出部２６１が、各中間テキストの動的な実行回数などを示すプロファイル情報を抽出すればよい。

【００５５】つまり、上述したプロファイル抽出部２６１と実行処理部２６２とにより、請求項２で述べた動的制御情報を作成し、この動的制御情報をアウトライン判定部２７０の実行回数判定部２７１に入力することにより、請求項２で述べた入力受付手段１２４の機能が実現されている。

【００５６】この場合は、この実行回数判定部２７１が、上述した動的制御情報で示された実行回数に応じて、サイズ判定部２４２およびレジスタ数判定部２４３の動作を制御すればよい。例えば、実行回数判定部２７１により、図７に示したステップ３２２の肯定判定に応じて、該当するインライン開始言の実行回数Nex が、所定の閾値Thn を超えているか否かを判定し、否定判定の場合にステップ３２４からステップ３２８の判定処理を実行すればよい。

【００５７】ここで、上述した所定の閾値Thn は、実験などで予め決定しておけばよい。また、コンパイルオプションとして、利用者が指定できるようにしても良い。この場合は、動的な実行回数に基づいて、サイズ判定部２４２およびレジスタ数判定部２４３を動作させることにより、請求項２で述べた第２評価手段１２５の機能を実現し、インライン展開の当否をより正当に評価することが可能である。

【００５８】したがって、インライン展開が真に効果的な関数呼び出しのみをインライン展開して、インライン展開による最適化効果を充分に引き出すことができ、効率的に作業を処理可能なオブジェクトコードを得ることができる。また、更に、上述したインライン展開過程で挿入した疑似言をアセンブラコードのでバッグ作業に利用することもできる。

【００５９】図１０に、請求項５のコンパイラ装置の実施例構成図を示す。図１０において、請求項５のコンパイラ装置は、図３に示したコンパイラ装置に置換処理部２８１を付加し、最適化後処理部２３０による処理結果に含まれる疑似言を後述するコメント文に置換して、バックエンド部４０５に送出する構成となっている。

【００６０】この置換処理部２８１は、インライン開始言とインライン終了言とを検出し、それぞれ該当する関数名を含んだコメント文（!Inline Begin 関数名）およびコメント文（!Inline End 関数名）に置換すればよい。

【００６１】これらのコメント文は、バックエンド部４０５により、そのままアセンブラソースとして出力されるから、これらのコメント文により、インライン展開された部分およびその関数名を示すことができる。上述したように、インライン処理部２１０で挿入された疑似言の入力に応じて、置換処理部２８１が動作することにより、請求項５で述べたコード付加手段１３１の機能を実現し、アセンブラソースにおいて、インライン展開された部分を示す情報を提供し、利用者のデバッグ作業に役立てることができる。

【００６２】この機能は、上述したように、インライン処理とバックエンド処理とにおいて実現されているから、アウトライン判定処理およびアウトライン処理を実行したか否かにかかわらず、インライン展開された部分についての情報を利用者に提供することができる。例えば、図１０に点線で示すように、コンパイルオプションに応じて、コンパイル制御部４０３が、最適化前処理部２２０および最適化後処理部２３０を制御して、最適化前処理部２２０による処理結果を直接に最適化後処理部２３０に送出すれば、アウトライン処理をスキップして、従来のコンパイラ装置と同等のコンパイル処理を行うことができる。

【００６３】この場合には、インライン展開した後の情報を最適化処理に活用することはできないが、翻訳に要する時間を従来のコンパイラ装置とほぼ同等とすることができ、また、上述した置換処理部２８１の動作により、インライン展開された部分に関する情報を含んだアセンブラソースを得ることができ、デバッグ作業に役立てることができる。

【００６４】

【発明の効果】以上説明したように、請求項１の発明によれば、インライン展開後に得られる静的な制御情報に基づいて、インライン展開によるメリットとデメリットとを比較評価し、効果的でないインライン展開部分を元の関数呼び出しに戻すことができるから、有効なインライン展開部分のみをオブジェクトコードへの変換処理に供することが可能である。これにより、インライン展開による最適化効果を充分に引き出して、処理効率の高いオブジェクトコードを得ることができるから、情報処理装置の処理能力の向上に大きく貢献することができる。

【００６５】同様に、請求項２の発明によれば、動的な制御情報に基づいて、真に有効なインライン展開部分を判別することにより、処理効率の高いオブジェクトコードを得ることができるから、特に実行頻度の高いプログラムのコンパイルに利用すれば、大きな処理能力向上効果が期待できる。また、請求項３の発明によれば、静的制御情報あるいは動的制御情報に加えて、制限付きの最適化処理による効果を考慮して、インライン展開部分をより精密に評価し、インライン展開による最適化効果を向上することができる。

【００６６】更に、請求項４の発明によれば、オブジェクトコードへの変換処理に先立って、インライン展開部分を含んだ中間テキストに対して、無制限の最適化処理を施すことにより、より処理効率の高いオブジェクトコードを得ることが可能となる。また、請求項５の発明によれば、オブジェクトコードにインライン展開部分を示す情報を残してデバッグ作業に供し、デバッグ作業を支援することができるから、プログラム開発者の作業負担を更に軽減することが可能である。

【図面の簡単な説明】

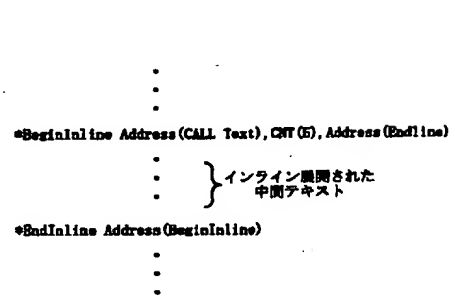
【図１】請求項１および請求項３のコンパイラ装置の原理ブロック図である。

【図２】請求項２および請求項４、５のコンパイラ装置の原理ブロック図である。

【図３】本発明のコンパイラ装置の実施例構成図であ

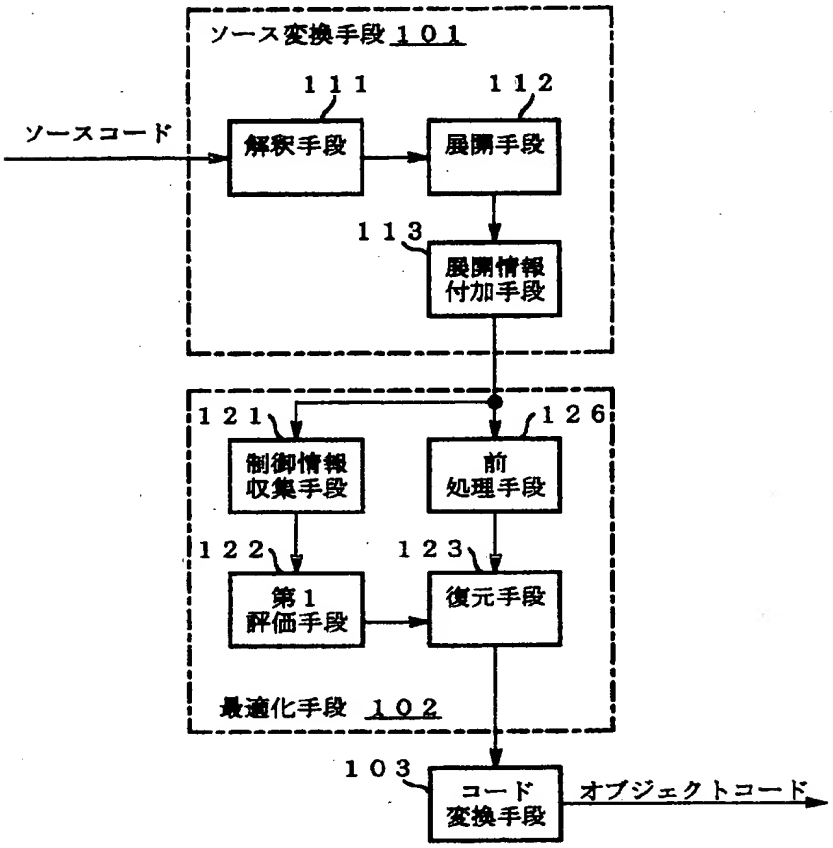
る。
【図４】コンパイル動作の概略を示す流れ図である。
【図５】疑似言挿入処理を説明する図である。
【図６】インライン展開動作を表す流れ図である。
【図７】アウトライン判定動作を表す流れ図である。
【図８】アウトライン処理動作を表す流れ図である。
【図９】請求項２のコンパイラ装置の実施例構成図である。
【図１０】請求項５のコンパイラ装置の実施例構成図である。
【図１１】従来のコンパイラ装置の構成例を示す図である。
【符号の説明】
１０１　ソース変換手段
１０２　最適化手段
１０３　コード変換手段
１１１　解釈手段
１１２　展開手段
１１３　展開情報付加手段
１２１　制御情報収集手段
１２２　第１評価手段
１２３　復元手段
１２４　入力受付手段
１２５　第２評価手段
１２６　前処理手段
１２７　後処理手段
１３１　コード付加手段
２１０、４０２　インライン展開部
２１１　受付処理部
２１２、２４２　サイズ判定部
２１３　疑似言挿入部
２１４　展開処理部
２１５　疑似言修飾部
２２０　最適化前処理部
２２１　基本処理部
２２２　データフロー収集部
２２３　制御フロー収集部
２３０　最適化後処理部
２４０、２７０　アウトライン判定部
２４１　ループ判定部
２４３　レジスタ数判定部
２４４　候補決定部
２４５　候補リスト
２５０　アウトライン処理部
２５１　候補検出部
２５２　削除処理部
２５３、２８１　置換処理部
２６１　プロファイル抽出部
２６２　実行処理部
２７１　実行回数判定部
４０１　フロントエンド部
４０３　コンパイル制御部
４０４　最適化処理部
４０５　バックエンド部
４０６　コード出力部

【図５】



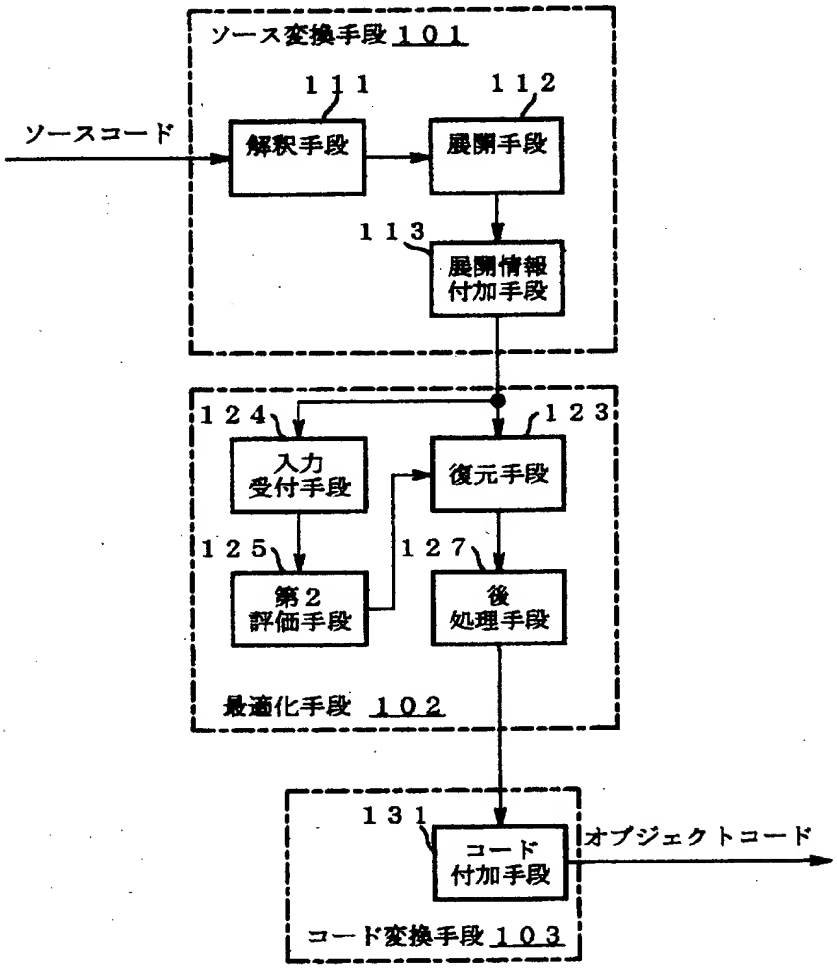
【図1】

請求項1および請求項3のコンパイラ装置の原理ブロック図



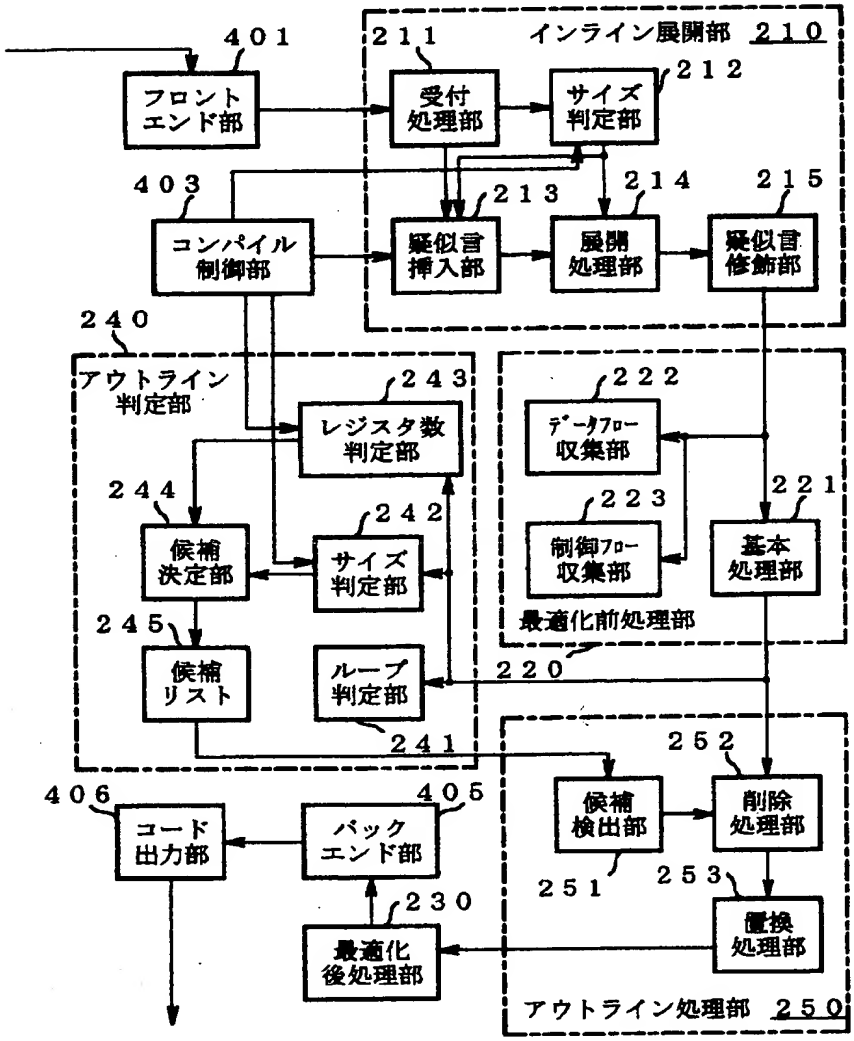
【図2】

請求項２および請求項４、５のコンパイラ装置の原理ブロック図



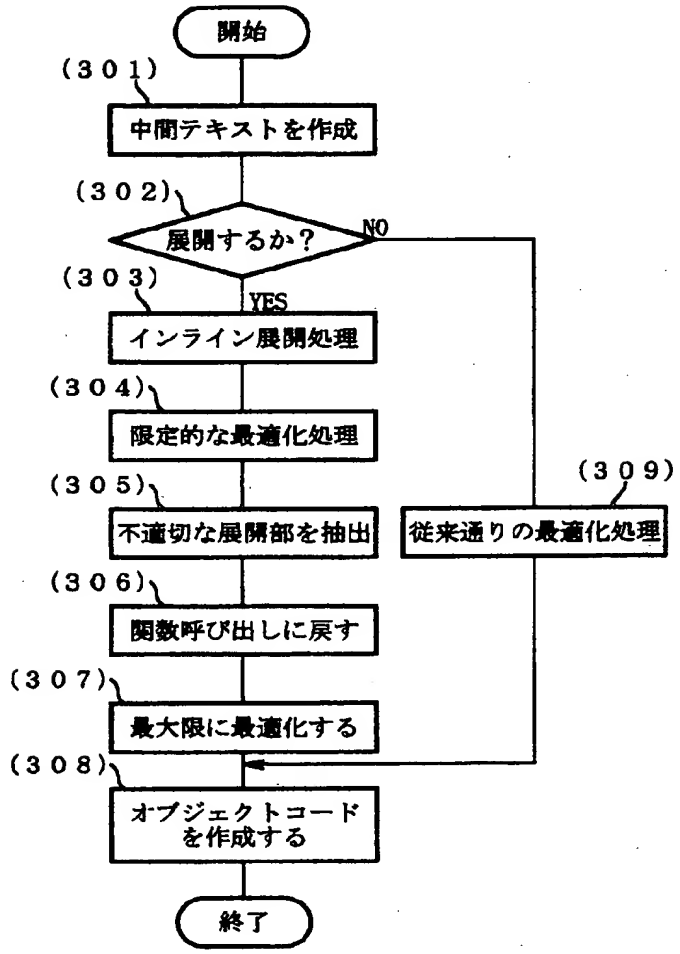
【図3】

本発明のコンパイラ装置の実施例構成図



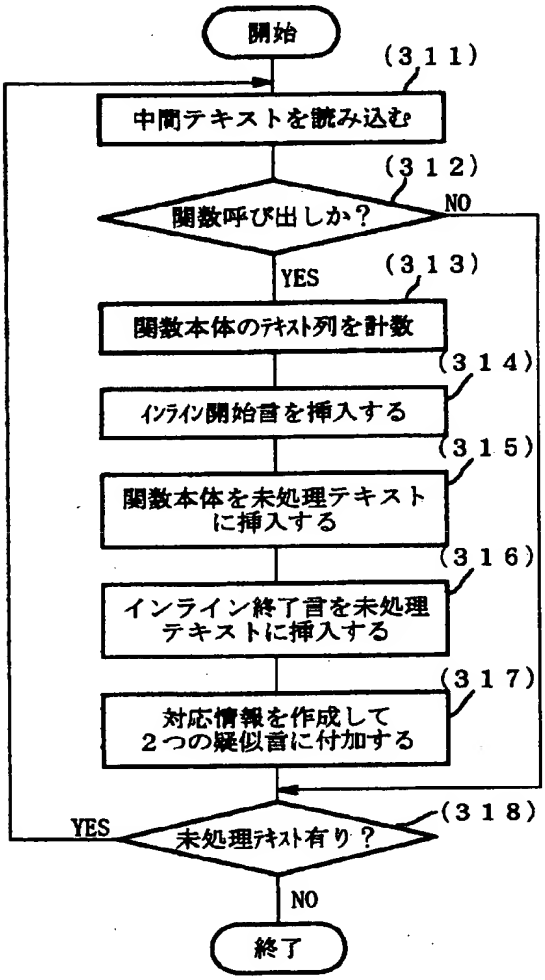
【図４】

コンパイル動作の概略を表す流れ図



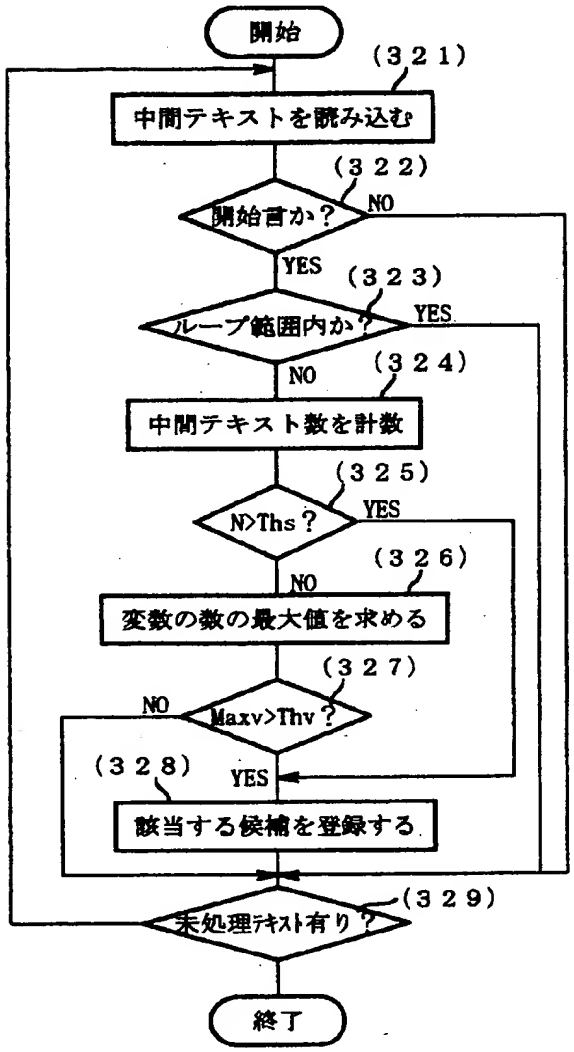
【図６】

インライン展開動作を表す流れ図



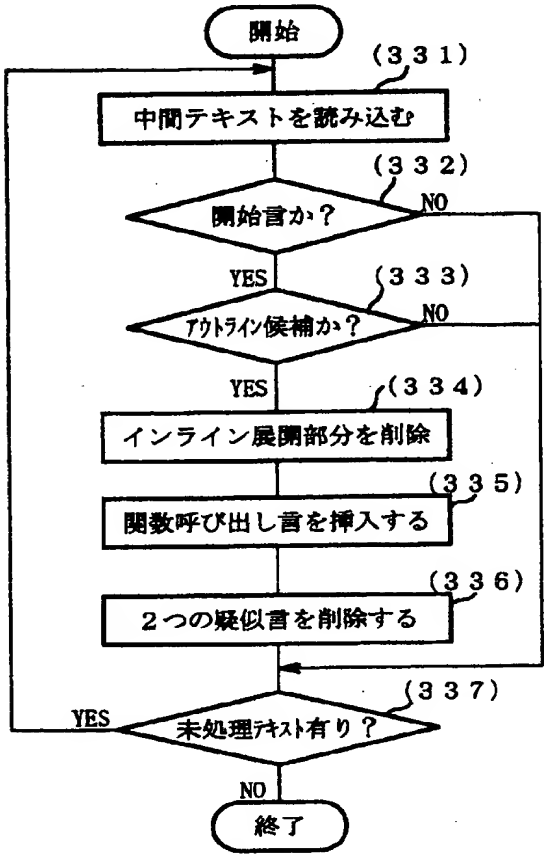
【図7】

アウトライン判定動作を表す流れ図



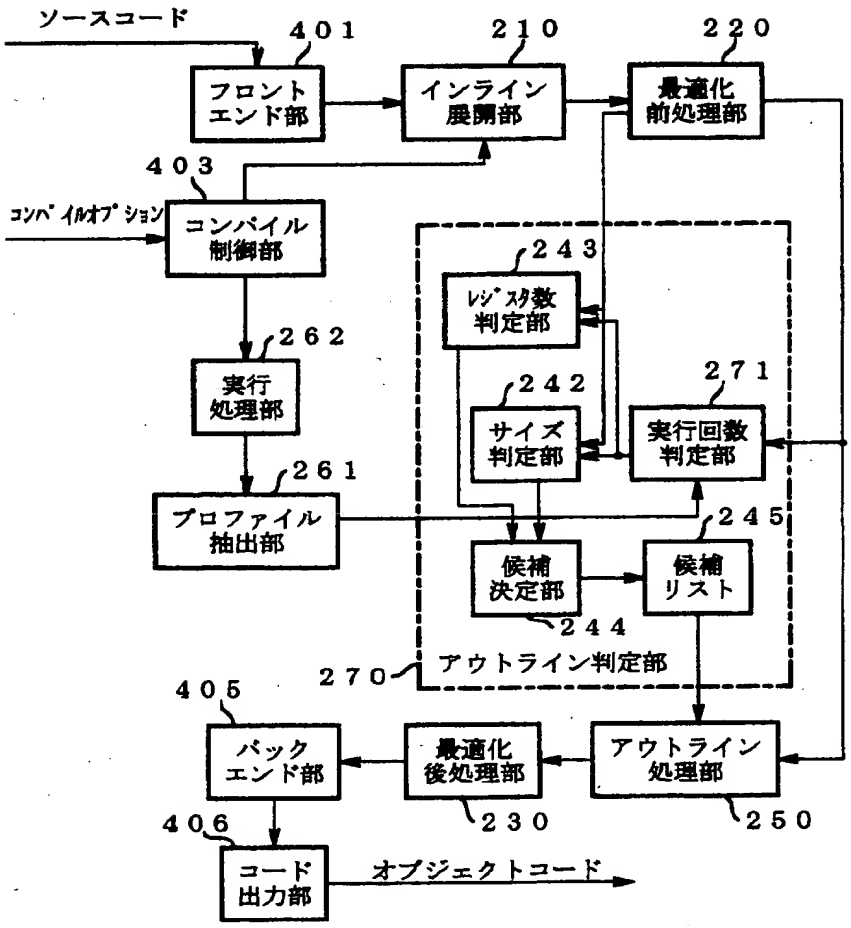
【図8】

アウトライン処理動作を表す流れ図



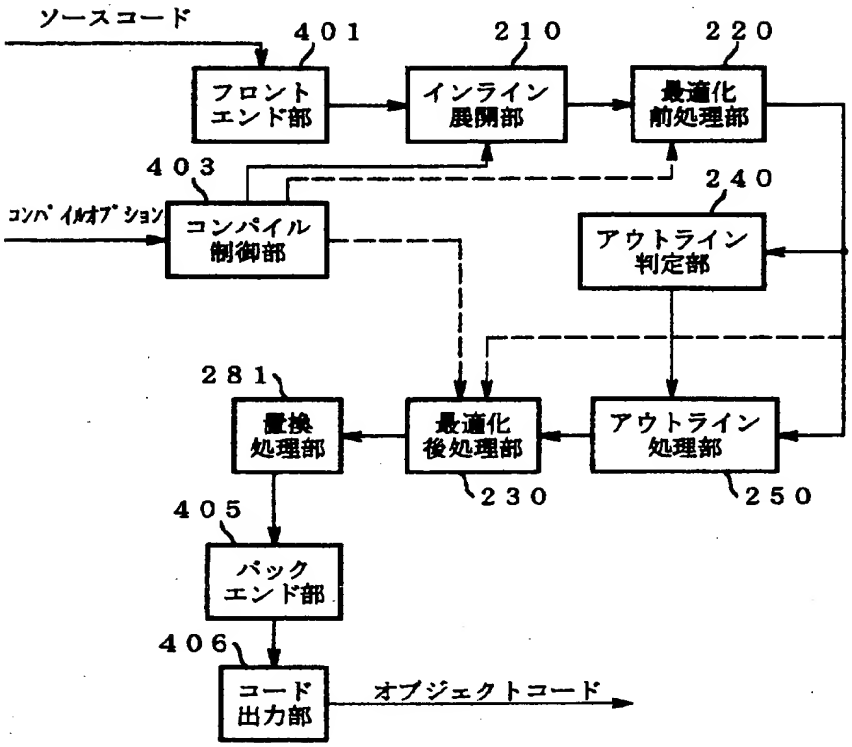
【図9】

請求項2のコンパイラ装置の実施例構成図



【図１０】

請求項５のコンパイラ装置の実施例構成図



【図１１】

従来のコンパイラ装置の構成例を示す図